1. Consider the following grammar: $S \rightarrow S \mid S \mid S \mid (S) \mid \text{true} \mid \text{false}$
   a. Compute First sets for each production and nonterminal
   b. Explain why the grammar cannot be parsed by a LL(1) parser

2. Consider the following grammar: $S \rightarrow abS \mid acS \mid c$
   a. Compute First sets for each production and nonterminal
   b. Show why the grammar cannot be parsed by a LL(1) parser.

3. Consider the following grammar: $S \rightarrow Sa \mid Sc \mid c$
   a. Show why the grammar cannot be parsed by a LL(1) parser.
   b. Rewrite the grammar so it can be parsed by a LL(1) parser.
   c. Using FIRST and FOLLOW, build the LL(1) table for the new grammar.
   d. Using the LL(1) table, show how a table-driven parser parses the string “caca”.

4. Consider the following grammar

<table>
<thead>
<tr>
<th>Goal</th>
<th>→</th>
<th>Expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expr</td>
<td>→</td>
<td>Term + Expr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Term – Expr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Term</td>
</tr>
<tr>
<td>Term</td>
<td>→</td>
<td>Factor * Term</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Factor / Term</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Factor</td>
</tr>
<tr>
<td>Factor</td>
<td>→</td>
<td>num</td>
</tr>
<tr>
<td></td>
<td></td>
<td>id</td>
</tr>
</tbody>
</table>

   a. Apply left factoring to create a LL(1) grammar
   b. Compute FIRST and FOLLOW for the modified grammar
   c. Using FIRST and FOLLOW, build the LL(1) table for the new grammar

5. Bottom up parsing
   a. Describe the difference between bottom-up and top-down parsing with respect to a grammar production $A \rightarrow \beta$
   b. Define handle
   c. Explain why bottom-up parsers result in right-most derivations in reverse.
   d. Given the following ACTION/GOTO table, perform a parse of “a”, assuming 0 is the start state. Show the contents of the stack, remaining input, and action performed at each step of the shift-reduce parse.
6. LR parsing
   a. What is the canonical set of LR(k) items for a grammar?
   b. What are ACTION/GOTO tables of an LR(k) parser for a grammar?
   c. Given an LR(k) item \([A \rightarrow \alpha \gamma, \delta]\) from a production \(A \rightarrow \beta\), describe the role of \(A, \alpha, \gamma, \delta\)
   d. Given an LR(k) item \([A \rightarrow \alpha \gamma, \delta]\), what lookahead allows a shift to be performed (ignoring any other LR(k) items added to the state by closure)?
   e. Given an LR(k) item \([A \rightarrow \alpha \gamma, \delta]\), what LR(k) items must be added to the closure?
   f. If the parser is in a state containing an LR(k) item \([A \rightarrow \alpha \gamma, \delta]\), what must be at the top of the stack, and why?
   g. Describe what constitutes a shift/reduce conflict
   h. Give an example of two LR(1) items causing a reduce/reduce conflict.
   i. Explain why there are no shift/shift conflicts

7. LALR parsing.
   Consider the following sets of LR(1) items in the states of a LR(1) parser.
   \[
   \begin{array}{c|c|c|c|}
   \text{State} & \text{Action} & \text{Goto} \\
   \hline
   0 & \text{shift 1} & \text{\$} & 2 & 3 \\
   1 & \text{shift 3} & \text{reduce} & A \rightarrow \epsilon & 0 \\
   2 & & \text{accept} & 3 & 0 \\
   3 & \text{shift 1} & \text{accept} & 3 & 1 \\
   \end{array}
   \]
   a. Find all shift/reduce and reduce/reduce conflicts. List the LR(1) items and lookaheads causing conflicts.
   b. What states would be merged in a LALR(1) parser?
   c. Are any new reduce/reduce conflicts introduced in the LALR(1) parser?
   d. Explain why LALR(1) parsers will not introduce new shift/reduce conflicts.
   e. What is the relationship between LALR(1) and SLR(1) parsers?
8. Consider the following grammar: $S \rightarrow \text{abS} \mid \text{acS} \mid c$
   a. Compute $\text{FOLLOW}(S)$
   b. Compute the canonical set of LR(0) items for the grammar
   c. Build the ACTION/GOTO table using LR(0) + $\text{FOLLOW}$
   d. Explain why the grammar is or is not SLR(1)
   e. Use the ACTION/GOTO table to parse the string “abc”

9. Consider the following grammar: $S \rightarrow \text{Sa} \mid \text{Sc} \mid c$
   a. Compute the canonical set of LR(1) items for the grammar
   b. Build the ACTION/GOTO table for the grammar
   c. Explain why the grammar is or is not LR(1)
   d. Use the ACTION/GOTO table to parse the string “caca”

10. Consider the following grammar: $S \rightarrow S + S \mid S * S \mid (S) \mid \text{num}$
    a. Compute the canonical set of LR(1) items for the grammar
    b. Build the ACTION/GOTO table for the grammar
    c. Explain why the grammar is or is not LR(1)
    d. Explain how to use the ACTION/GOTO table so that
        i. * has higher precedence than + and *, + are both left associative
        ii. * has higher precedence than +, * is left associative, and + is right associative
        iii. + has higher precedence than * and *, + are both right associative

11. Consider the following grammar: $S \rightarrow AS \mid b \quad A \rightarrow SA \mid a$
    a. Compute the canonical set of LR(0) items for the grammar
    b. Build the ACTION/GOTO table for the grammar
    c. Compute the canonical set of LR(1) items for the grammar
    d. Build the ACTION/GOTO table for the grammar

12. Consider the following grammar: $S \rightarrow S ; a \mid a$
    a. Is the grammar SLR(1)? Is it LR(1)?

13. Consider the following grammar: $S \rightarrow aAd \mid bBd \mid aBe \mid bAe \quad A \rightarrow c \quad B \rightarrow c$
    a. Is the grammar LR(1)? Is it LALR(1)?

14. Consider the following grammar: $S \rightarrow Aa \mid bAc \mid Bc \mid bBa \quad A \rightarrow d \quad B \rightarrow d$
    a. Is the grammar LR(1)? Is it LALR(1)?